Programmable Bricks: Toys to think with

Article *in* IBM Systems Journal · February 1996

DOI: 10.1147/sj.353.0443 · Source: IEEE Xplore

4 authors, including:

Mitchel Resnick
Massachusetts Institute of Technology
132 PUBLICATIONS **19,353** CITATIONS

SEE PROFILE

Randy Sargent
Carnegie Mellon University
33 PUBLICATIONS **875** CITATIONS

SEE PROFILE

Programmable Bricks: Toys to think with

by M. Resnick
F. Martin
R. Sargent
B. Silverman

In this paper, we discuss the applications and implications of the Programmable Brick—a tiny, portable computer embedded inside a LEGO® brick, capable of interacting with the physical world in a large variety of ways. We describe how Programmable Bricks make possible a wide range of new design activities for children, and we discuss experiences in using Programmable Bricks in three types of applications: autonomous creatures, active environments, and personal science experiments.

In many educational computer projects, children control and manipulate worlds that exist in the computer. Using one program, children can control the movements of Newtonian "dynaturtles" in physics microworlds.[1] Using a different program, children can create and manipulate simulated urban worlds, constructing houses, roads, and factories, and setting tax rates for the city.[2]

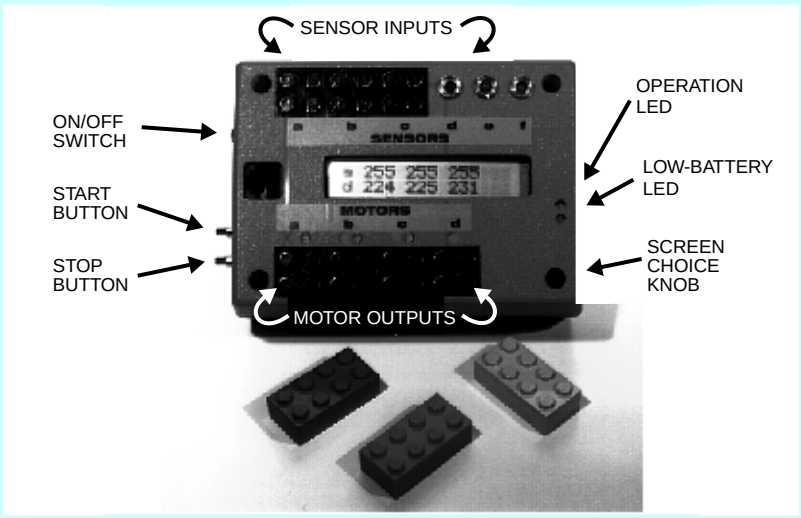
But instead of controlling and manipulating *worlds in the computer*, what if children could control and manipulate *computers in the world*? That is, what if children could spread computation throughout their own personal worlds? For example, a child might attach a tiny computer to a door, then program the computer to make lights turn on automatically whenever anyone enters the room. Or the child might program the computer to greet people as they enter the room—or to sound an alarm if anyone enters the room at night.

In this paper, we describe a new technology, called the Programmable Brick (Figure 1), that makes such activities possible, and we explore how this new technology might open new learning opportunities for children. The Programmable Brick is a tiny, portable computer embedded inside a pocket-sized LEGO** brick. The brick is capable of interacting with the physical world in a large variety of ways (including sensors and infrared communication). The Programmable Brick makes possible a wide range of new design activities for children, encouraging children to see themselves as designers and inventors. At the same time, we believe that these activities could fundamentally change how children think about (and relate to) computers and computational ideas.

Ubiquitous computing

Our work on the Programmable Brick fits within an area of research sometimes known as "ubiquitous computing." This research aims to change the nature of computing in very fundamental ways. As described in one research article: "We live in a complex world, filled with myriad objects, tools, toys, and people. Our lives are spent in diverse interaction with this environment. Yet, for the most part, our computing takes place sitting in front of, and staring at, a single glowing screen attached to an array of buttons and a mouse."[3] Ubiquitous computing, by contrast, aims at "integrating computers seamlessly into the world at large."[4] The goal is to spread computation throughout

Figure 1 The Programmable Brick



the environment, embedding computation in all types of objects and artifacts.

For example, at some laboratories, people are starting to wear "active badges" so that the building can "know" the location of all people in the building at all times. At other sites, researchers are using "smart whiteboards"[5,6] or "digital desks"[7] that keep track of everything that is written on them. To some, ubiquitous computing means that computation "disappears," becoming totally integrated into everyday objects. We take a somewhat broader view of ubiquitous computing, including new artifacts like "personal digital assistants" and hand-held computers that enable people to access and exchange information wherever they are.

Our work on the Programmable Brick is resonant with these efforts to distribute computational power, but it differs along several important dimensions:

- *The Programmable Brick is designed for children.* Most work on ubiquitous computing is aimed at adults, particularly adults in business settings. For example, most personal digital assistants are designed for tasks like keeping track of appointments and downloading stock quotes. Those activities are not of great interest to children. In developing the Programmable Brick, we considered how we could make ubiquitous-computing activities meaningful to the lives of children.
- *The Programmable Brick gives users the power to create and control.* In many ubiquitous-computing activities, the roles of designers and users are separate and distinct. Designers create ubiquitous-computing devices (like active badges and smart whiteboards) and users interact with them. Our goal is to blur this distinction, giving much greater control to users so that they can create their own ubiquitous-computing activities. The Programmable Brick is explicitly *programmable* so that users can continually modify and customize its behavior. In this way, the Programmable Brick fits clearly within a constructionist approach to learning.[8]
- *The Programmable Brick provides rich connections to the world.* Many ubiquitous-computing activities focus on transfer of information (such as downloading airline schedules). We are more interested in connecting computation to physical objects, enabling people to program computers to sense the world around them and to perform actions in response. Toward that end, the Programmable Brick

has a rich assortment of input/output capabilities, including ten ports for motors and sensors, and built-in speaker and infrared communications.

LEGO/Logo

The Programmable Brick project extends our previous work with LEGO/Logo.[9,10] LEGO/Logo links the popular LEGO construction kit with the Logo programming language. In using LEGO/Logo, children start by building machines out of LEGO pieces, using not only the traditional LEGO building bricks but newer pieces like gears, motors, and sensors. Then they connect their machines to a computer and write computer programs (using a modified version of Logo) to control the machines. For example, a child might build a LEGO house with lights and program the lights to turn on and off at particular times. Then, the child might build a garage and program the garage door to open whenever a car approaches.

Logo itself was developed in the late 1960s as a programming language for children.[11,12] In the early years, the most popular use of Logo involved a "floor turtle," a simple mechanical robot connected to the computer by a long "umbilical cord." With the proliferation of personal computers in the late 1970s, the Logo community shifted its focus to "screen turtles." Screen turtles are much faster and more accurate than floor turtles, and thus allow children to create and investigate more complex geometric effects.

In some ways, LEGO/Logo might seem like a throwback to the past, since it brings the turtle off the screen and back into the world. But LEGO/Logo differs from the early Logo floor turtles in several important ways. First of all, LEGO/Logo users are not given ready-made mechanical objects; they build their own machines before programming them. Second, children are not restricted to turtles. Elementary-school students have used LEGO/Logo to build and program a wide assortment of creative machines, including a programmable pop-up toaster, a "chocolate-carob factory" (inspired by the Willy Wonka children's stories), and a machine that sorts LEGO bricks according to their lengths. The LEGO company now sells a commercial version of LEGO/Logo. It is used in more than a dozen countries, including more than 15000 elementary and middle schools in the United States.

LEGO/Logo has some limitations. For one thing, LEGO/Logo machines must be connected to a desktop computer with wires. Wires are a practical nuisance, particularly when children use LEGO/Logo to create mobile "creatures." Wires get tangled with other objects in the environment, they get twisted in knots as the creature rotates, and they restrict the overall range of the creature. Wires are also a conceptual nuisance. It is difficult to think of a LEGO/Logo machine as an autonomous creature as long as it is attached by umbilical cord to a computer.

Members of our research group have tried to solve these problems in several ways. We experimented with various technologies for wireless communication, to get around the problem of wires. But none of these approaches satisfied us. So we decided to make a more serious modification: we began to build electronics *inside* the LEGO bricks. We have taken several approaches. The "Braitenberg Brick" system, developed primarily by Fred Martin with inspiration from the book *Vehicles*,[13] is based on a set of low-level "logic bricks" (such as and-gates, flip-flops, and timers). Children can create different behaviors by wiring these bricks together in different ways.[14,15]

It is difficult to think of a machine as an autonomous creature if it is attached by umbilical cord to a computer.

Programmable Bricks

The Braitenberg Bricks have dedicated functions. The flip-flop brick, for instance, has a very specialized function: it holds one bit of state, and it changes that state whenever it receives a sharp transition in its input. But why should we be restricted to dedicated bricks? Why not put a full computer in a LEGO brick?

That is what we have done in the Programmable Brick project. In designing the Programmable Brick, we had several overarching goals. Each goal involved some type of "multiplicity":

- *Multiple activities.* We wanted the Programmable Brick to support a wide variety of different activi-

Figure 2 Fourth-grade students test the behaviors of their Programmable Brick "creature"

ties—so that it could connect to the interests and experiences of a wide variety of people. While some people might use the Brick to create their own scientific instruments, others might use it to create their own musical instruments.

- *Multiple input/output modalities.* We wanted the Programmable Brick to connect to many things in the world. To do that, the Brick needed many different types of output devices (such as motors, lights, beepers, infrared transmitters) and many different types of input devices (such as touch sensors, sound sensors, light sensors, temperature sensors, infrared receivers). Indeed, the number of possible applications of the Brick expands greatly with each new input or output device, since each new device can be used in combination with all of the others.
- *Multiple processes.* Children working on LEGO/Logo projects often want to control two or more things at the same time. For example, they might want to make a Ferris wheel and merry-go-round turn in synchrony, while a song plays in the background and an electric eye automatically counts the rotations of the rides. With standard programming languages, it is very difficult to achieve this effect: the user must explicitly interleave the multiple threads of control. In the Programmable Brick, we wanted to support parallel processing, so that users could easily write programs to control multiple outputs and check multiple sensors all at the same time.
- *Multiple bricks.* We wanted Programmable Bricks not only to act on their own but to interact with one another. In that way, children could program Bricks to share sensor data with each other, or they could create "colonies" of interacting creatures. These types of activities would enable children to explore the scientific ideas of emergence and self-organization.[16]

Based on these goals, we developed the Programmable Brick shown in Figure 1. About the size of a child's juice box, the Programmable Brick is based on the Motorola 6811 processor with 32 kilobytes of nonvolatile random access memory, and it has a wide variety of input-output possibilities. The Brick can control four motors or lights at a time, and it can receive inputs from six sensors. The Brick supports infrared communications (with a built-in infrared receiver and an attachable transmitter). The Brick includes a two-line liquid-crystal display, plus a knob and two buttons for interacting directly with the Brick. An earlier version of the Brick had many fewer input-output features. (See References 17 and 18 for more details.)

To program the Programmable Brick, you first write programs on a standard personal computer (using a special version of Logo known as Brick Logo), then download the programs via a cable to the Programmable Brick. Then you can disconnect the cable and take the brick with you. The programs remain stored on the brick. When you want to execute a program on the brick, you can scroll through a menu of programs on the two-line liquid crystal display screen (using the knob to scroll), then press a button to run the selected program. (For more technical information about the Programmable Brick, see our World Wide Web site on the Internet.[19])

Experiences with the Brick

In our early experiments with the Programmable Brick, we observed three broad categories of applications: active environments, autonomous creatures, and personal science experiments. In this section, we discuss example projects in each category.

Active environments. One of the earliest projects with the Programmable Brick involved two children. Andrew and Dennis, ages 11 and 12, were intrigued

with the idea of making an "active environment"—making the environment "come alive" and react to people. After some consideration, they decided to make a device to flip on a room's light switch when people entered the room, and flip it off when people left.

At first, Andrew and Dennis studied the different possible sensors, trying to figure out which one could be connected to the door, and how. They decided to try a "bend" sensor to sense the opening of the door. (The bend sensor is a plastic whisker, several inches long, that gives a measure of how much it is bent.) They first tried to mount the sensor to the wall approximately where the doorstop was, but then decided that people would need to open the door very wide before the sensor detected anything. Then, they tried mounting the sensor at the door hinge in such a way that the sensor was bent in proportion to how widely the door was opened.

Before programming, Andrew and Dennis tested the value of the bend sensor at different positions of the door, to find out if they had mounted the sensor well and if the sensor would really give them the information they wanted. Then, they built a LEGO mechanism to flip the light switch on the wall of the room. The mechanism connected a motor, through a gear train, to a lever that pushed against the light switch. They designed their mechanism in such a way that spinning the motor one way would turn the light on, while the reverse direction would turn the light off.

At this point, Andrew and Dennis started focusing on the algorithm for flipping the light switch when the door opened. They realized there was a problem: the door sensor indicated when the door was opened, but it did not tell whether people were entering or exiting the room. The children wanted some sort of sensor to tell whether someone was entering the room (in which case their machine should turn on the light) or leaving the room (in which case the machine should turn off the light).

After a little thinking, Andrew and Dennis came up with a clever solution: they attached a LEGO bar to the door handle on the inside of the door, and connected a LEGO touch sensor to this bar. In this way, the Programmable Brick could tell if people were leaving (in which case the door would be open and the touch sensor in the handle would be pressed), or if people were entering (the door would be open but no signal would come from the touch switch).

Figure 3 Three sixth-grade students show off their LEGO dinosaur, including a knapsack to carry its Programmable Brick

Once the second sensor was in place and tested, Andrew and Dennis wrote a simple program that continuously checked both sensors, and flipped the light switch depending on the sensor values. Once they got the project working, they ran in and out of the room repeatedly, breaking into big smiles each time the lights switched on and off.

Autonomous creatures. With traditional LEGO/Logo technology and LEGO constructions tethered to the computer, children are more likely to think of building "machines" (like elevators and Ferris wheels) rather than "creatures" (like robotic dinosaurs). As noted earlier, the dangling wires are both a practical nuisance and a conceptual nuisance. The Programmable Brick removes these problems, freeing children to think about new types of autonomous-creature projects.

In the first large-scale use of Programmable Bricks outside of a laboratory setting, a group of schools in Rhode Island used the Bricks to create a "Robotic Park" exhibition. Students created robotic "animals" that participated in a type of artificial 4-H show (Figures 2 and 3). The animal theme was chosen in an effort to make the project appealing to a broad range

Figure 4 Programmable Brick attached to handlebars of Eric's bicycle

of students, while also connecting to important ideas from the science curriculum. As students built and programmed their LEGO creatures, they studied how "real" animals live and behave, and they applied their findings to their LEGO constructions.

In one fourth-grade class, students (working in teams of three or four) built a robotic crab, a turtle, and an alligator. In a fifth-grade class, students built an "anchovy fish" and a dinosaur. In all cases, students made their LEGO creatures mimic the behaviors of actual animals. For example, the LEGO crab had a pair of pincers that started snapping when the crab ran into something; the LEGO turtle's head retracted when its nose was bumped; the LEGO dinosaur was attracted to flashes of light (like the dinosaur in Jurassic Park[20]).

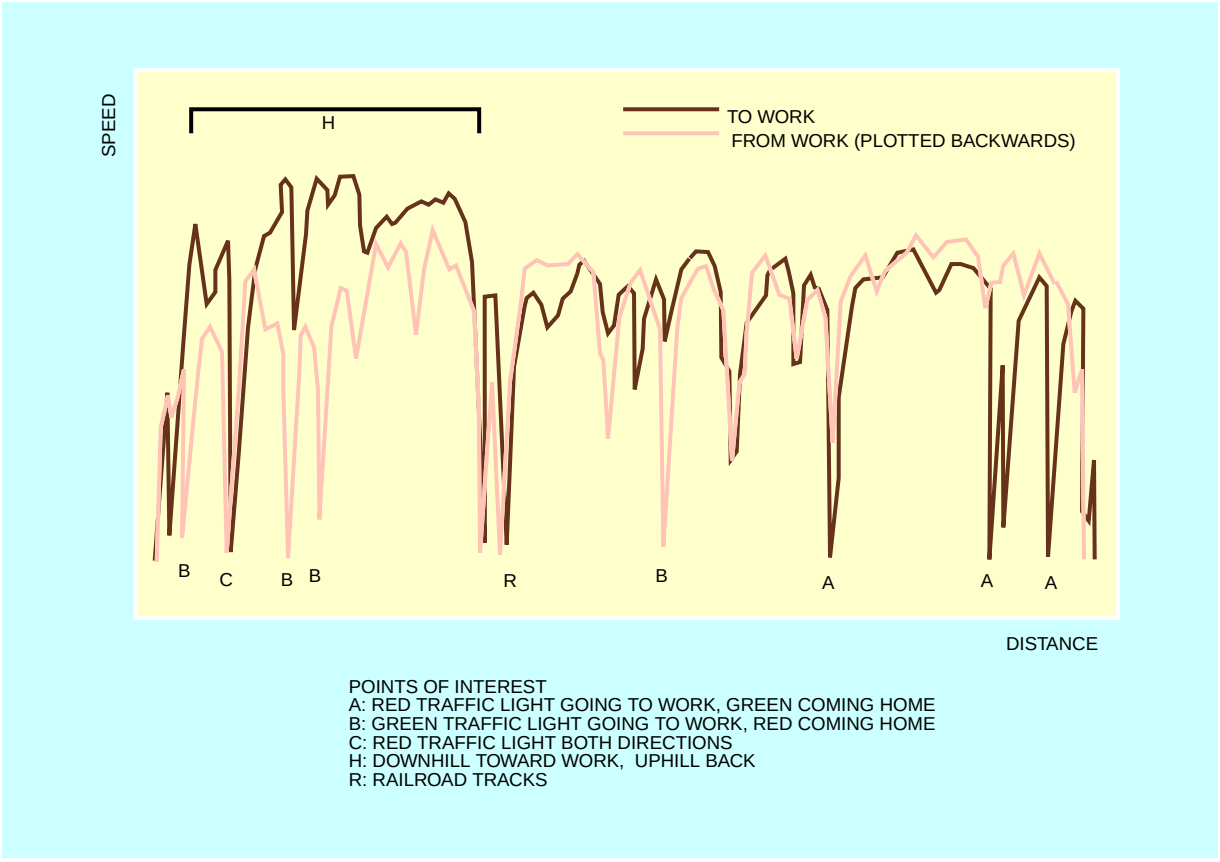
Many of the creatures were equipped with five or six different sensors. The multiprocessing capabilities of the Programmable Brick were critical in many cases. Students tended to write multiple "condition-action" rules to connect sensor stimuli to behavioral reactions. For example, one student wrote a short piece of Logo code telling a creature to back up and turn left when the right-hand touch sensor was pressed, and another short piece of code telling the creature to back up and turn right when the left-hand touch sensor was pressed. While these rules were active, the student added more rules telling the creature how to respond to inputs from various light sensors. In this way, it was easy for students to develop complex behaviors for their creatures. (See Reference 21 for a discussion of university-level students working on similar projects.)

Another group of students (ages 12 to 16) worked on similar projects at a four-day workshop at the Boston Museum of Science. One focus of this workshop was the use of multiple processes for multiple behaviors. The Programmable Brick's software includes primitives that allow students to turn on and off the different processes from program control. In one project, Darryl created two procedures: one to make the creature follow a light, the other to make the creature avoid obstacles. Darryl made both procedures run at the same time. The parallelism of the Programmable Brick was very useful: the creature could check for obstacles even as it followed the light. But when the creature detected an obstacle, there were problems. As the obstacle-avoidance behavior tried to guide the creature around the obstacle, the light-following behavior kept pointing the creature back toward the light. Darryl solved this problem by modifying the obstacle-avoidance behavior to temporarily turn off the light-following behavior while the creature was navigating around a detected obstacle.

Personal science experiments. We believe that the Programmable Brick will make possible new types of science experiments, in which children investigate everyday phenomena in their lives (both in and out of the classroom). For example, children could attach Programmable Bricks (and related sensors) to their own bodies to monitor and analyze how their legs move while they are running. We believe that students are much more likely to make deep connections to scientific thinking and scientific ideas when they use computation in this new way, continually designing and redesigning their own personal investigations.

Brian Silverman and his son Eric used a Programmable Brick to conduct one such investigation. Eric attached a Programmable Brick to the handlebars of his bicycle (Figure 4) and programmed the Brick to collect data from a sensor on the front bicycle wheel. After trying various sensors to measure the rotation of the wheel, Brian and Eric settled on a magnet mounted to the wheel, and a reed relay (mechanical magnetic sensor) mounted next to the wheel. Once per rotation of the wheel, the sensor would detect the magnet. The brick was programmed to record, every two seconds, the speed of the wheel and the number of wheel rotations.

Figure 5 Speed versus distance riding bike to and from work



Later, Brian attached the Brick to his own bicycle in order to investigate the speed of his bicycle during his daily bike trips commuting to and from work. To graph the results, Brian and Eric wrote a special-purpose program. Although the recorded data measured speed vs time, they decided to graph speed vs distance instead. With this graph, they could superimpose trips taken on different days. Events at the same location on each trip (like traffic lights or stop signs) would be at the same place along the x-axis of the graph. The graph is quite striking: it shows things like train tracks (where Brian had to slow down every day) and traffic lights (where he came to a stop only some of the time). See Figure 5 for the graph of a single trip to work and back.

By plotting return trips in reverse, and superimposing them, many of the features (such as slowing down for the train tracks) were held in common. But consistent discrepancies (one stretch of the trip consistently being faster in one direction than in the other) indicated something was different between the trips to and from—for example, an uphill or downhill slope.

There are already many "bike computers" on the market to help people monitor their cycling. Why use a Programmable Brick? The Programmable Brick allows users much greater flexibility in collecting and analyzing data. For example, the Programmable Brick enabled Brian and Eric to develop a new representation (speed vs distance) that commercial bike computers would never support.

Things That Think (and make us think)

The MIT Media Laboratory recently initiated a major new research project called "Things That Think." The overarching goal is to embed computational capabili-

ties in everyday objects like furniture, shoes, and toys. The Programmable Brick fits within this new initiative—but with an important twist. In our work, we are interested in things that think, not because they might accomplish particular tasks more cheaply or easily or intelligently, but because they might enable people to think about things in new ways. That is, things that think are most interesting to us when they also act as "things to think with." We believe that Programmable Bricks act in just that way, enabling children to perform new types of explorations and experiments and to engage in new types of thinking.

"Things that think" are more interesting when they also act as "things to think with."

The Programmable Brick project is just beginning; we have many ideas for future extensions. We are currently developing a new set of bricks, known as Crickets, that are much smaller and lighter than the original Programmable Bricks. Each Cricket is not much bigger than the 9-volt battery that powers it. A Cricket can control two motors and receive information from two sensors, and it comes with built-in two-way infrared communications capabilities for "talking" with other Crickets and other electronic devices. The small size of the Crickets, along with their enhanced communication capabilities, opens up new possibilities for applications. People might wear the Crickets to exchange information with one another, as in the Thinking Tags project (described elsewhere in this issue[22]). Or children might use the Crickets to experiment with decentralized and self-organizing phenomena, creating a whole "colony" of ant-like mobile robots that interact with one another.

There are still many obstacles to the widespread use of Programmable Bricks and Crickets—especially in school settings. The major obstacles are not technological, but in the structure and organization of schools themselves. Most interesting Programmable Brick projects require extended blocks of time; they cannot be squeezed into standard 50-minute class sessions or standard two-week curriculum units. Moreover, Programmable Brick projects typically cut across disciplinary boundaries, and they often engage students in thinking about ideas (such as feedback loops) that are not traditionally included in precollege curricula. As a result, educators are not quite sure where to "fit" Programmable Brick activities. For the Programmable Brick to become successful in school settings, we need to work with educators to find ways around these constraints. Indeed, one of the most important effects of new technologies like the Programmable Brick is that they can provoke us to rethink some of our basic assumptions about education.

Acknowledgments

Seymour Papert, Steve Ocko, and Allan Toft have provided encouragement, inspiration, and ideas for the Programmable Brick project. Andrew Blumberg, Yuying Chen, Fei Hai Chua, Dennis Evangelista, Chris Gatta, Hanna Jang, Owen Johnson, Mark Neri, Brian Robertson, Victor Tsou, and Elaine Yang all contributed to the development effort. The LEGO Group and the National Science Foundation (Grants 9153719-MDR, 8751190-MDR, and RED-9358519) have provided financial support. Portions of this paper previously appeared in Reference 23.

For more information on the Programmable Brick, see our World Wide Web site on the Internet.[19]

Appendix: Twenty things to do with a Programmable Brick

More than 20 years ago, as researchers and educators were just beginning to explore the possibilities of computers in education, Seymour Papert and Cynthia Solomon wrote a memo called "Twenty Things to Do with a Computer."[24] The memo described a wonderful collection of activities, pushing computers in directions that few other people had imagined. Some of the activities on their list eventually became commonplace; others are still visionary today. A few years later, Danny Hillis (then an undergraduate at MIT) wrote a memo entitled "Ten Things to Do with a Better Computer,"[25] describing a new set of activities that would be possible if computers could execute instructions in parallel. Hillis later realized some of these ideas in his massively parallel Connection Machine computer.[26]

In the same spirit, we have compiled a new list entitled "Twenty Things to Do with a Programmable Brick."[10]

1. Create a "haunted house." Attach a Programmable Brick to the door to make creaking sounds whenever the door is opened. Program another Brick to drop spiders on people when they walk through the door. Build a LEGO platform for a pumpkin, and program a Brick to drive the pumpkin around the room.
2. Connect sensors to various parts of your body. Then program a Programmable Brick to monitor your heartbeat and breathing as you walk and run. Or program the Brick to play different sounds when you move different parts of your body.
3. Take a Programmable Brick with you to measure the pH level of the water in local streams, or the noise levels at a local construction site.
4. Create a LEGO musical instrument. The instrument might have buttons like a flute, or a sliding part like a trombone, or a completely new interface that you invent. Start by writing a simple program so that the Programmable Brick plays different notes (or melodies) when you move different parts of the instrument. Then enhance the program so that the Brick improvises on your notes. Or program the Brick to play "rounds" (by playing a second copy of your notes with a delay).
5. Put a Programmable Brick and light sensor on the door to keep track of the number of people that enter the room. Then program the Brick to greet people as they enter the room (with music or digitized speech).
6. Set up a weather station on the roof of the building.
7. Use a Programmable Brick to find out if the light really does go off when you shut the refrigerator door.
8. Attach a Programmable Brick to an ashtray, and program it to play a coughing sound whenever anyone uses the ashtray.
9. Build a remote-controlled LEGO car. Use a standard television remote control to communicate (via infrared transmission) with a Programmable Brick in the car.
10. Create an "intelligent room" that automatically turns on the lights when someone walks in the room. (Here's one approach. Build a LEGO machine that turns on the light switch, and connect it to a Programmable Brick. Use another Programmable Brick to detect when anyone enters the room. Use infrared transmission to communicate between the two Bricks.)
11. Use a Programmable Brick to control a videocamera (via infrared transmission). Program the Brick to make a time-lapse video of a plant growing (taking a few frames every hour or day).
12. Use a Programmable Brick to program your VCR.
13. Send secret messages across the room to someone else who also has a Programmable Brick.
14. Put a Brick on your dog's collar and collect data about your dog's behavior. How much time does your dog spend running around? Discuss whether experimenting on your dog is ethical.
15. Use a Brick to record your dog barking. Then put the Brick in a remote-controlled LEGO car. Play the barking sound when the LEGO car gets near a cat. How does the cat react?
16. Build a LEGO creature that you can interact with. Program the creature to act in different ways when you clap once, or clap twice, or shine a light in its "eyes."
17. Build a LEGO creature that explores its environment. Program the creature to find the part of the room with the most light or the highest temperature. Next, put a plant on your LEGO creature, so that the plant will always move to the part of the room with the most light (or the highest temperature). Use other sensors to monitor the growth of the plant.
18. Build a LEGO machine that can water your plants, then program a Brick to make the machine water the plants every few days.
19. Create a game where each player carries a Programmable Brick. Program the Bricks so that they give instructions to the players, and send messages from one player to another.
20. Think up 20 more things to do with a Programmable Brick.

**Trademark or registered trademark of LEGO Systems, Incorporated.

Cited references

1. H. Abelson and A. diSessa, *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*, MIT Press, Cambridge, MA (1980).
2. W. Wright, *SimCity*, Maxis, Orinda, CA (1990).
3. P. Wellner, W. Mackay, and R. Gold, "Computer Augmented Environments: Back to the Real World," *Communications of the ACM* **36**, No. 7, 24–26 (July 1993).
4. M. Weiser, "The Computer for the 21st Century," *Scientific American* **265**, No. 3, 94–104 (September, 1991).
5. S. Elrod, R. Bruce, R. Gold, D. Goldberg, F. Halasz, W. Janssen, D. Lee, K. McCall, E. Pedersen, K. Pier, J. Tang, and B. Welch, "Liveboard: A Large Interactive Display Supporting Group Meetings, Presentations, and Remote Collaborations," *Proceedings, ACM Conference on Human Factors in Computing Systems*, ACM Press, New York (1992), pp. 599–607.
6. M. Weiser, "Some Computer Science Issues in Ubiquitous

Computing," *Communications of the ACM*, **36** No. 7, 25–26 (July, 1993).
7. P. Wellner, "Interacting with Paper on the Digital Desk," *Communications of the ACM* **36**, No. 7, 87–96 (July, 1993).
8. S. Papert, "Situating Constructionism," I. Harel and S. Papert, Editors, *Constructionism*, Ablex Publishing, Norwood, NJ (1991).
9. M. Resnick, S. Ocko, and S. Papert, "LEGO, Logo, and Design," *Children's Environments Quarterly* **5**, No. 4, 14–18 (1988).
10. M. Resnick, "Behavior Construction Kits," *Communications of the ACM* **36**, No. 7, 64–71 (July 1993).
11. S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas*, Basic Books, New York (1980).
12. B. Harvey, *Computer Science Logo Style*, MIT Press, Cambridge, MA (1985).
13. V. Braitenberg, *Vehicles*, MIT Press, Cambridge, MA (1984).
14. N. Granott, "Puzzled Minds and Weird Creatures: Spontaneous Inquiry and Phases in Knowledge Construction," I. Harel and S. Papert, Editors, *Constructionism*, Ablex Publishing, Norwood, NJ (1991).
15. D. Hogg, F. Martin, and M. Resnick, "Braitenberg Creatures," *Epistemology and Learning Memo 13*, MIT Media Laboratory, Cambridge, MA (1991).
16. M. Resnick, *Turtles, Termites, and Traffic Jams*, MIT Press, Cambridge, MA (1994).
17. F. Martin, *Children, Cybernetics, and Programmable Turtles*, master's degree thesis, MIT Media Laboratory, Cambridge, MA (1988).
18. M. Bourgoin, "Children Using LEGO Robots to Explore Dynamics," I. Harel, Editor, *Constructionist Learning*, MIT Media Laboratory, Cambridge, MA (1990).
19. The World Wide Web site for the Programmable Brick is at http://el.www.media.mit.edu/groups/el/projects/programmable-brick/.
20. M. Crichton, *Jurassic Park*, Alfred A. Knopf, New York (1990). The 1993 movie based on the book was directed by Steven Spielberg.
21. F. Martin, *Circuits to Control: Learning Engineering by Designing LEGO Robots*, Ph.D. dissertation, MIT Media Laboratory, Cambridge, MA (1994).
22. R. Borovoy, M. McDonald, F. Martin, and M. Resnick, "Things That Blink: Computationally Augmented Name Tags," *IBM Systems Journal* **35**, Nos. 3&4, 488–495 (1996, this issue).
23. R. Sargent, M. Resnick, F. Martin, and B. Silverman, "Building and Learning with Programmable Bricks," Y. Kafai and M. Resnick, Editors, *Constructionism in Practice*, Lawrence Erlbaum, Mahwah, NJ (1996).
24. S. Papert and C. Solomon, "Twenty Things to Do with a Computer," *Artificial Intelligence Memo 248*, MIT Artificial Intelligence Laboratory, Cambridge, MA (1971).
25. W. D. Hillis, "Ten Things to Do with a Better Computer," unpublished memo available from MIT Artificial Intelligence Laboratory, Cambridge, MA (1975).
26. W. D. Hillis, *The Connection Machine*, MIT Press, Cambridge, MA (1985).

Accepted for publication April 4, 1996.

Mitchel Resnick *MIT Media Laboratory, 20 Ames Street, Cambridge, Massachusetts 02139-4307 (electronic mail: mres@media.mit.edu).* Dr. Resnick, an associate professor at the MIT Media Laboratory, studies the role of new technological tools in learning and education. He has helped develop a variety of "computational construction kits" (including LEGO/Logo and StarLogo), and he cofounded the Computer Clubhouse, an afterschool learning center for youth from under-served communities. He earned a B.A. in physics at Princeton University, and M.S. and Ph.D. degrees in computer science at MIT. He won a National Science Foundation Young Investigator Award in 1993, and he is author of the book *Turtles, Termites, and Traffic Jams*, published by MIT Press. Dr. Resnick is on the Board of Overseers and is chair of the education committee at The Computer Museum.

Fred Martin *MIT Media Laboratory, 20 Ames Street, Cambridge, Massachusetts 02139-4307 (electronic mail: fredm@media.mit.edu).* Dr. Martin earned a B.S. degree in computer science in 1986, an M.S. in mechanical engineering in 1988, and a Ph.D. in media arts and sciences in 1994, all from the Massachusetts Institute of Technology. His doctoral dissertation explored learning in an intensive, design-rich robot-building class he codeveloped for MIT undergraduates. Dr. Martin's research interests include: the role of experiential knowledge in learning formal scientific and engineering methods; design-rich environments for learning; and robots as a medium for exploring engineering practice. He is presently a research scientist with the Epistemology and Learning Group at the MIT Media Laboratory.

Randy Sargent *Newton Research Labs, 14813 NE 13th Street, Bellevue, Washington 98007 (electronic mail: rsargent@newton-labs.com).* Mr. Sargent holds a bachelor's degree in computer science and a master's degree in media arts and sciences, both from MIT. He was a founding organizer of the MIT robot design competition, and he was centrally involved in the development of the Programmable Brick. He is a cofounder of Newton Research Labs, a software company developing products for robotics hobbyists and researchers.

Brian Silverman *Logo Computer Systems Inc., P.O. Box 162, Highgate Springs, Vermont 05460 (electronic mail: brian@lcsi.ca).* Mr. Silverman is Director of Research at Logo Computer Systems Inc. (LCSI) and a visiting scientist at the MIT Media Lab. He was one of the founders of LCSI, the world's leading developer of Logo software. He has directed the development of more than a dozen commercial educational software products (including LogoWriter, MicroWorlds, and the Phantom Fishtank), many of which have won major awards from industry groups and publications. At MIT, he has been centrally involved in the development of StarLogo, Programmable Bricks, and Crickets.

Reprint Order No. G321-5616.